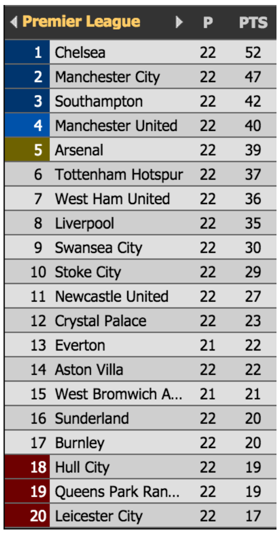
NGUYEN Trong Khoa - NGUYEN Ngoc Chau Sang

NETWORK I 2014 - HOMEWORK

PART I

1.1.1 Description

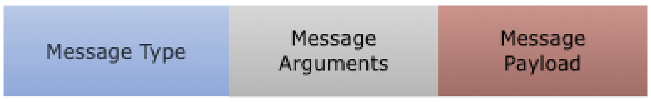
In this part, we implemented a client capable of scraping data from `http://www.livescore.com/` to get the league tables of Premier League, Serie A, Liga BBVA, Bundesliga, and Ligue 1. Each league table contains a certain number of football teams, each football team has the information of ranking, number of played match and point. The following image is an example for one league table:



	Premier League	P	PTS
1	Chelsea	22	52
2	Manchester City	22	47
3	Southampton	22	42
4	Manchester United	22	40
5	Arsenal	22	39
6	Tottenham Hotspur	22	37
7	West Ham United	22	36
8	Liverpool	22	35
9	Swansea City	22	30
10	Stoke City	22	29
11	Newcastle United	22	27
12	Crystal Palace	22	23
13	Everton	21	22
14	Aston Villa	22	22
15	West Bromwich A...	21	21
16	Sunderland	22	20
17	Burnley	22	20
18	Hull City	22	19
19	Queens Park Ran...	22	19
20	Leicester City	22	17

After scraping data, we store these data into a small database. Our database implemented as a TCP server. We also defined our own application protocol for the communication between the client and the server.

1.1.2 Message structure



Each message has 3 parts:

- Message type: to distinguish messages with different semantics. In this project, we have 5 types:
  - **UPDATE**: is sent when client requests to store the league tables into server. In this case, the MESSAGE PAYLOAD contains content of league tables in JSON format
  - **SELECT**: is sent when client requests to query the league tables.
  - **CLOSE**: is sent when client requests to close the connection
  - **RES_UPDATE**: is send when server replies the UDPATE message. The MESSAGE PAYLOAD contains the status of UPDATE request:
    - OK
    - FAIL
  - **RES_SELECT**: is send when server replies the SELECT message. The MESSAGE PAYLOAD contains the content of the league tables in JSON format
  - **RES_CLOSE**: is send when server accepts the CLOSE message. The MESSAGE PAYLOAD contains the status of CLOSE request:
    - OK

- Message arguments: information that complements the message type with additional semantics. We save this field for the future requirements (in this project, we did not need to use this field)
- Message payload: data in arbitrary length

To implement the message structure, we used the message syntax as following:

TYPE LEN ARGS\nDATA

in which:

- type: one or more alphanumeric characterss
- len: one or more decimal digits
- args: zero or more non-whitespace characters separated by a space
- data: data in arbitrary length

1.1.3 Desgin for concurrent

We used multi-threads for implementing our concurrent server. The main thread takes responsible of accepting new connection from clients, then creating one thread per client. In this part, we used "multi-threads" instead of "threads pool" because the number of connection is small, so creating one thread per client is not considered to be expensive in this scenario.

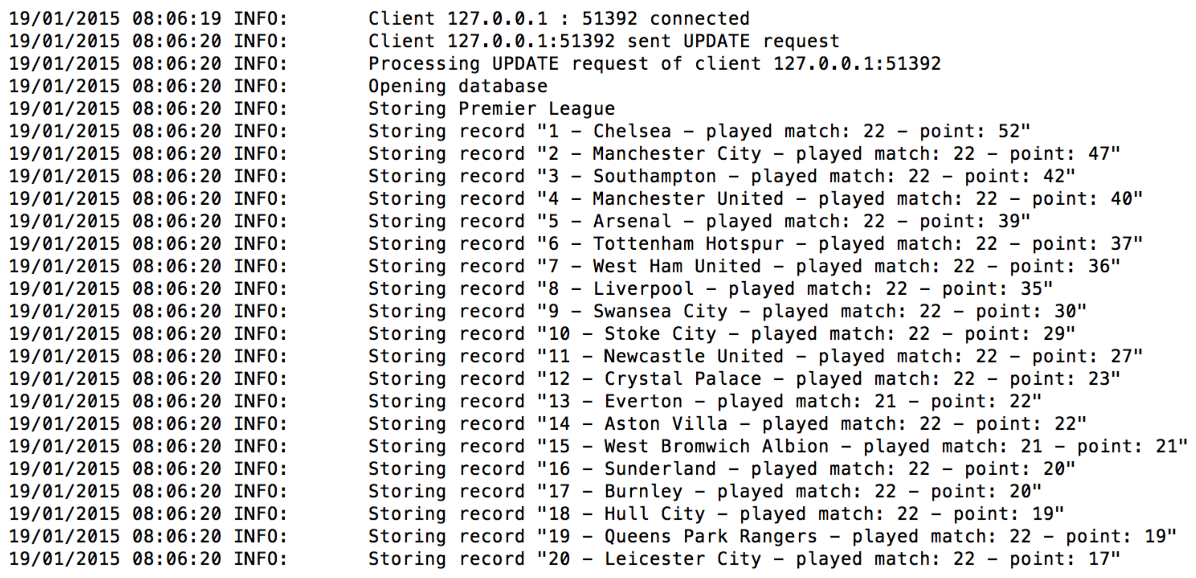
1.1.4 Handling simultaneous access to server's "databse"

Our "database" in the server side is a text file with the data in JSON format. Each time the client requests UPDATE or SELECT, the server opens text file to WRITE or READ data. So there will be a case in that many clients request UPDATE or SELECT simultaneously to the server. That the reason, we used "threading.Lock" to make READ and WRITE operations to be atomic (it means no more one thread can READ or WRITE at the same time)

1.1.5 Logging

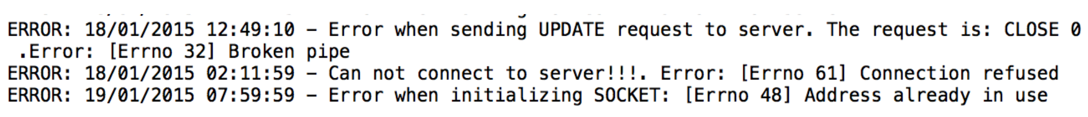
All the normal operations (new connection, receive requests, process requests, store record...) of the server are logged to console. The detail of errors or failures are logged to "server.log" for further debugging and security reasons

Console log:



```
19/01/2015 08:06:19 INFO:       Client 127.0.0.1 : 51392 connected
19/01/2015 08:06:20 INFO:       Client 127.0.0.1:51392 sent UPDATE request
19/01/2015 08:06:20 INFO:       Processing UPDATE request of client 127.0.0.1:51392
19/01/2015 08:06:20 INFO:       Opening database
19/01/2015 08:06:20 INFO:       Storing Premier League
19/01/2015 08:06:20 INFO:       Storing record "1 - Chelsea - played match: 22 - point: 52"
19/01/2015 08:06:20 INFO:       Storing record "2 - Manchester City - played match: 22 - point: 47"
19/01/2015 08:06:20 INFO:       Storing record "3 - Southampton - played match: 22 - point: 42"
19/01/2015 08:06:20 INFO:       Storing record "4 - Manchester United - played match: 22 - point: 40"
19/01/2015 08:06:20 INFO:       Storing record "5 - Arsenal - played match: 22 - point: 39"
19/01/2015 08:06:20 INFO:       Storing record "6 - Tottenham Hotspur - played match: 22 - point: 37"
19/01/2015 08:06:20 INFO:       Storing record "7 - West Ham United - played match: 22 - point: 36"
19/01/2015 08:06:20 INFO:       Storing record "8 - Liverpool - played match: 22 - point: 35"
19/01/2015 08:06:20 INFO:       Storing record "9 - Swansea City - played match: 22 - point: 30"
19/01/2015 08:06:20 INFO:       Storing record "10 - Stoke City - played match: 22 - point: 29"
19/01/2015 08:06:20 INFO:       Storing record "11 - Newcastle United - played match: 22 - point: 27"
19/01/2015 08:06:20 INFO:       Storing record "12 - Crystal Palace - played match: 22 - point: 23"
19/01/2015 08:06:20 INFO:       Storing record "13 - Everton - played match: 21 - point: 22"
19/01/2015 08:06:20 INFO:       Storing record "14 - Aston Villa - played match: 22 - point: 22"
19/01/2015 08:06:20 INFO:       Storing record "15 - West Bromwich Albion - played match: 21 - point: 21"
19/01/2015 08:06:20 INFO:       Storing record "16 - Sunderland - played match: 22 - point: 20"
19/01/2015 08:06:20 INFO:       Storing record "17 - Burnley - played match: 22 - point: 20"
19/01/2015 08:06:20 INFO:       Storing record "18 - Hull City - played match: 22 - point: 19"
19/01/2015 08:06:20 INFO:       Storing record "19 - Queens Park Rangers - played match: 22 - point: 19"
19/01/2015 08:06:20 INFO:       Storing record "20 - Leicester City - played match: 22 - point: 17"
```

Server.log:

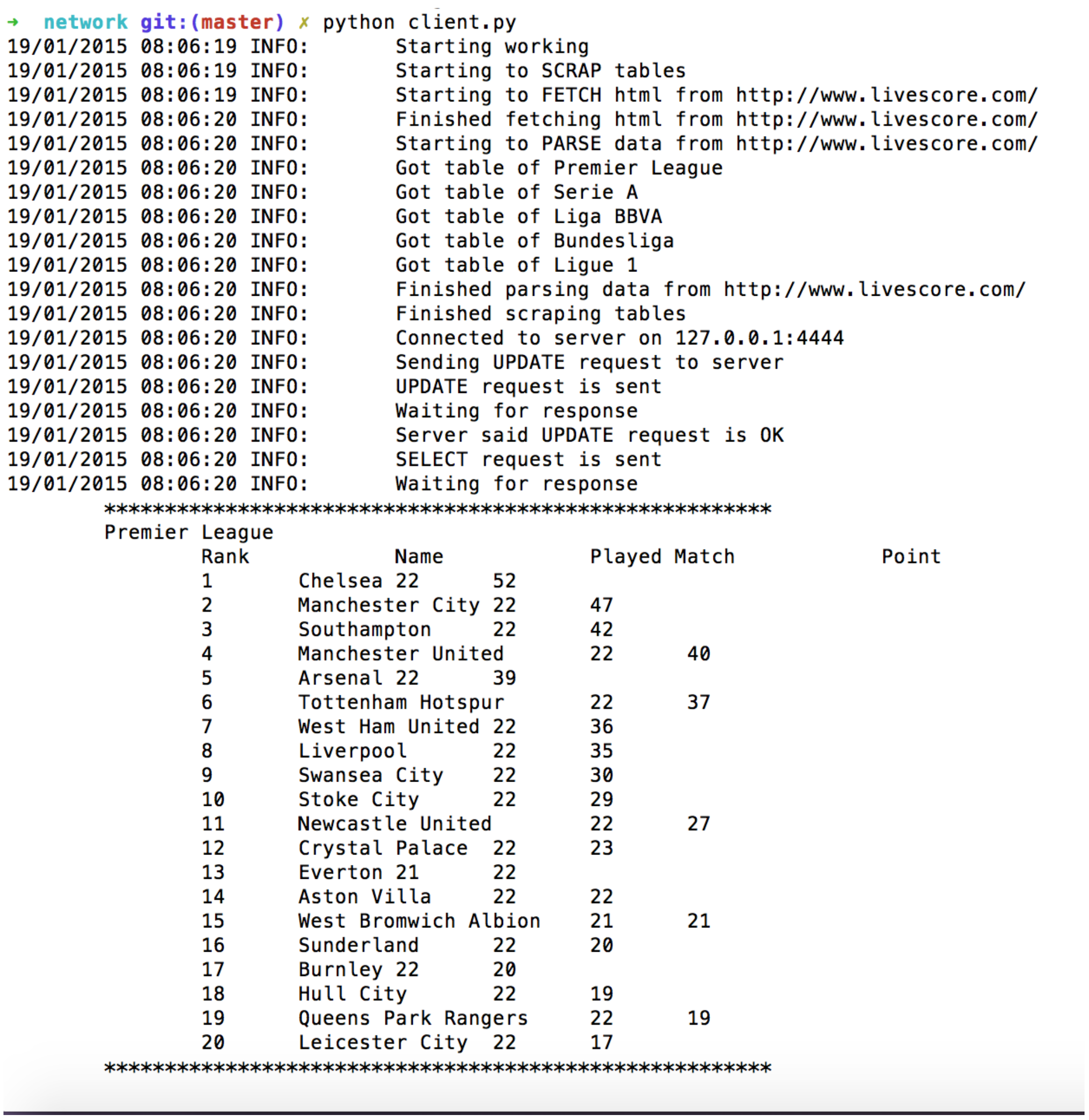


```
ERROR: 18/01/2015 12:49:10 - Error when sending UPDATE request to server. The request is: CLOSE 0
 .Error: [Errno 32] Broken pipe
ERROR: 18/01/2015 02:11:59 - Can not connect to server!!!. Error: [Errno 61] Connection refused
ERROR: 19/01/2015 07:59:59 - Error when initializing SOCKET: [Errno 48] Address already in use
```

1.1.6 Instruction for running server and client

- Run **python server.py** to start the server. After starting, the server is ready for new connections from clients.
- Run **python client.py** to start one instance of client. After starting, the client will start to scrap data, then connect and send UPDATE request to server to store data. After that, it requests SELECT to quey data, and finally it requests CLOSE to close the connection.

  Working period of client side:



```
➜  network git:(master) ✗ python client.py
19/01/2015 08:06:19 INFO:       Starting working
19/01/2015 08:06:19 INFO:       Starting to SCRAP tables
19/01/2015 08:06:19 INFO:       Starting to FETCH html from http://www.livescore.com/
19/01/2015 08:06:20 INFO:       Finished fetching html from http://www.livescore.com/
19/01/2015 08:06:20 INFO:       Starting to PARSE data from http://www.livescore.com/
19/01/2015 08:06:20 INFO:       Got table of Premier League
19/01/2015 08:06:20 INFO:       Got table of Serie A
19/01/2015 08:06:20 INFO:       Got table of Liga BBVA
19/01/2015 08:06:20 INFO:       Got table of Bundesliga
19/01/2015 08:06:20 INFO:       Got table of Ligue 1
19/01/2015 08:06:20 INFO:       Finished parsing data from http://www.livescore.com/
19/01/2015 08:06:20 INFO:       Finished scraping tables
19/01/2015 08:06:20 INFO:       Connected to server on 127.0.0.1:4444
19/01/2015 08:06:20 INFO:       Sending UPDATE request to server
19/01/2015 08:06:20 INFO:       UPDATE request is sent
19/01/2015 08:06:20 INFO:       Waiting for response
19/01/2015 08:06:20 INFO:       Server said UPDATE request is OK
19/01/2015 08:06:20 INFO:       SELECT request is sent
19/01/2015 08:06:20 INFO:       Waiting for response
        *******************************************************
        Premier League
                Rank            Name            Played Match            Point
                1       Chelsea 22      52
                2       Manchester City 22      47
                3       Southampton     22      42
                4       Manchester United       22      40
                5       Arsenal 22      39
                6       Tottenham Hotspur       22      37
                7       West Ham United 22      36
                8       Liverpool       22      35
                9       Swansea City    22      30
                10      Stoke City      22      29
                11      Newcastle United        22      27
                12      Crystal Palace  22      23
                13      Everton 21      22
                14      Aston Villa     22      22
                15      West Bromwich Albion    21      21
                16      Sunderland      22      20
                17      Burnley 22      20
                18      Hull City       22      19
                19      Queens Park Rangers     22      19
                20      Leicester City  22      17
        *******************************************************
```

PART II

PROJECT LINK

- The first part of the project is available at: `https://github.com/nncsang/NetworkI-Homework`
- The second part of the project is available at: `https://github.com/nncsang/NetworkI-Homework`